Comparing Java Web Frameworks

JSF, Spring MVC, Stripes, Struts 2, Tapestry and Wicket

Matt Raible
matt@raibledesigns.com
http://raibledesigns.com
Today's Agenda

- Introductions
- Pros and Cons
- Smackdown
- Conclusion
- Q and A
Introductions

- Your experience with webapps?
- Your experience with Java EE?
- What do you want to get from this session?
- Experience with Maven, Tomcat, Hibernate, Spring?
- Web Framework Experience:
  - Spring MVC, Struts 2, Stripes, JSF, Tapestry, Wicket
Who is Matt Raible?

- Power user of Java Open Source Frameworks
- Author of Spring Live and Pro JSP 2.0
- Founder of AppFuse and AppFuse Light
- Member of Java EE 5, JSF 1.2 and Bean Validation Expert Groups
- Committer on Apache Projects: Roller and Struts
- Java Blogger since 2002
My Experience
Pros and Cons

I've got a new way for you to build web apps!
JSF

Pros:
- Java EE Standard - lots of demand and jobs
- Fast and easy to develop with initially
- Lots of component libraries

Cons:
- Tag soup for JSPs
- Doesn't play well with REST or Security
- No single source for implementation
Spring MVC

Pros:
- Lifecycle for overriding binding, validation, etc.
- Integrates with many view options seamlessly: JSP/JSTL, Tiles, Velocity, FreeMarker, Excel, PDF
- Inversion of Control makes it easy to test

Cons:
- Configuration intensive - lots of XML
- Almost too flexible - no common parent Controller
- No built-in Ajax support
Stripes

Pros:
- No XML - Convention over Configuration
- Good documentation (easy to learn)
- Enthusiastic community

Cons:
- Small Community
- Not as actively developed as other projects
- Hard-coded URLs in ActionBeans
Struts 2

Pros:
- Simple architecture - easy to extend
- Tag Library is easy to customize with FreeMarker or Velocity
- Controller-based or page-based navigation

Cons:
- Documentation is poorly organized
- No feedback for missing properties or invalid OGNL expressions
- Googling results in Struts 1.x documentation
Tapestry

Pros:
- Very productive once you learn it
- Templates are HTML - great for designers
- Lots of innovation between releases

Cons:
- Documentation very conceptual, rather than pragmatic
- Steep learning curve
- Long release cycles - major upgrades every year
Wicket

Pros:
- Great for Java developers, not web developers
- Tight binding between pages and views
- Active community - support from the creators

Cons:
- HTML templates live next to Java code
- Need to have a good grasp of OO
- The Wicket Way - everything done in Java
The Smackdown
Evaluation Criteria

- **Ajax Support:** Is it built-in and easy to use?
- **Bookmark-ability:** Can users bookmark pages and return to them easily?
- **Validation:** How easy is it to use and does it support client-side (JavaScript) validation?
- **Testability:** How easy is it to test Controllers out of container?
Evaluation Criteria, cont.

- **Post and Redirect**: How does the framework handle the duplicate post problem?
- **Internationalization**: How is i18n supported and how easy is it to get messages in Controllers?
- **Page Decoration**: What sort of page decoration/composition mechanisms does the framework support?
- **Community and Support**: Can you get questions answered quickly (and respectfully)?
Evaluation Criteria, cont.

- **Tools**: Is there good tool (particularly IDE) support for the framework?

- **Marketability of Skills**: If you learn the framework, will it help you get a job?

- **Job Count**: What is the demand for framework skills on dice.com and indeed.com?
Ajax Support

- Is Ajax support built-in and easy to use?
  - JSF: No Ajax support, use ICEfaces and Ajax4JSF
  - Stripes: No libraries, supports streaming results
  - Struts 2: Dojo built-in, plugins for GWT, JSON
  - Spring MVC: No libraries, use DWR & Spring MVC Extras
  - Tapestry: Dojo built-in in 4.1
  - Wicket: Dojo and Script.aculo.us (Wicket Stuff)
Bookmarking and URLs

- JSF does a POST for everything - URLs not even considered
- Stripes uses conventions, but you can override
- Struts 2 has **namespaces** - makes it easy
- Spring MVC allows **full URL control**
- Tapestry still has somewhat ugly URLs
- Wicket allows pages/URLs to be **mounted**
Validation

- JSF has ugly default messages, but easiest to configure
- Spring MVC allows you to use Commons Validator - a mature solution
- Struts 2 uses OGNL for powerful expressions - client-side only works when specifying rules on Actions
- Tapestry has very robust validation - good messages without need to customize
- Stripes and Wicket do validation in Java - no client-side
Testability

- Spring and Struts 2 allow easy testing with mocks (e.g. EasyMock, jMock, Spring Mocks)
- Tapestry appears difficult to test because page classes are abstract, Creator class simplifies
- JSF page classes can be easily tested and actually look a lot like Struts 2 actions
- Wicket has WicketTester, a powerful solution
- Stripes has Servlet API Mocks and MockRoundtrip
Post and Redirect

- The duplicate-post problem: redirect after POST
- Is there support for allowing success messages to live through a redirect?
  - Spring MVC allows you to add parameters to a redirect
  - Stripes, Tapestry and Wicket all have "flash" support
  - Struts 2 requires a custom solution
  - JSF requires a custom solution, i18n messages difficult to get in page beans
Internationalization

- JSTL's `<fmt:message>` tag makes it easy
- No standard for getting i18n messages in controller classes
- Stripes, Spring MVC and JSF use a single `ResourceBundle` per locale
- Struts 2, Tapestry and Wicket advocate separate files for each page/action
- JSF requires resource bundle to be declared on each page
- Tapestry's `<span key="key.name">` is awesome
Page Decoration

- Tiles Experience: used since it first came out
- SiteMesh is much easier to setup and use
- Tiles can be used in Struts 2, Spring and JSF
  
    Requires configuration for each page
- SiteMesh can be used with all frameworks
  
    Requires very little maintenance after setup
- SiteMesh not supported or recommended for use with JSF, Tapestry or Wicket
Marketability of Skills

- Struts 1 is still in high-demand and widely-used
- Spring is getting more press, but mostly due to the framework’s other features
- JSF is becoming popular; awful with JSP
- Struts 2 is gaining ground, but very scarce on job boards
- Tapestry has increased in popularity in last couple years
- Wicket and Stripes are virtually unknown
Pretty Graphs
Dice.com Job Count

November 14, 2007

- Struts 2: 77
- Spring MVC: 98
- Stripes: 20
- JSF: 15
- Wicket: 69
- Tapestry: 748
Dice Job with Struts 1.x
Job Trends

Graph showing job trends from April 2005 to September 2007. The graph plots the percentage of matching job postings for various technologies over time. The technologies include Struts2, Webwork, Spring MVC, Stripes Java, JSF Java, Tapestry Java, and Wicket Java. The data is sourced from indeed.com.
Job Trends vs. Struts
Job Trends vs. J2EE
Job Trends vs. Java

[Graph showing job trends over time for various Java technologies, with the x-axis representing time from April 2005 to September 2007, and the y-axis representing the percentage of matching job postings. The graph includes lines for struts2, webwork, spring mvc, stripes java, jsf java, tapestry java, wicket java, j2ee, and java.]
Employer Search on Monster.com
Resumes posted in last 2 weeks

- Struts 2: 592
- Spring MVC: 75
- Stripes: 84
- Wicket: 49
- JSF: 11
- Tapestry: 12
Mailing List Traffic

* Spring MVC is not listed here because they have a forum instead of a mailing list and I couldn’t figure out a way to count the number of messages for each month.
Releases in 2007

- Struts 2: 6
- Spring MVC: 1
- Stripes: 2
- MyFaces: 2
- Wicket: 2
- Tapestry: 4
Tools Available

- Struts 2
- Spring MVC
- Stripes
- Wicket
- JSF
- Tapestry

Bar chart showing tools available in May 2007 with counts:
- Struts 2: 4
- Spring MVC: 3
- Stripes: 3
- Wicket: 12
- JSF: 7
- Tapestry: unknown
Books on Amazon

October 2007

<table>
<thead>
<tr>
<th>Struts 2</th>
<th>Spring MVC</th>
<th>Stripes</th>
<th>JSF</th>
<th>Wicket</th>
<th>Tapestry</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
How do you choose?
Eliminate, Don’t Include
6 Important Factors

- What type of Application are you building?
- Ease of Development / Is full-stack an option?
- Project Community
- Project Future and Roadmap
- Maintenance
- Technical Features
Don’t believe the Hype

Don’t believe blogs and articles
Try it yourself
Believe developers, not evangelists
Believe developers that are experienced with the framework and have used it in production
Beware of corporate interests - they can twist marketing
Books are a good sign
Best Tool for the Job

- Frameworks have sweet spots - is your application one of them?
- Pick 2-3 frameworks for your type of application...
- ... and prototype!
- If prototyping is painful, switch
- Make sure you prototype more than one and do a presentation comparing the pros and cons of each
After Choosing...

- Document the reasons for your decision
- Allow developers to challenge it
- Allow your prototype to be written with other frameworks
- Don’t be afraid to try new frameworks
- Don’t be afraid to use old frameworks
- Don’t be afraid to keep your existing framework
What do I think?
Conclusion

- The future is bright because of all the competition
- Developers should know more than one web framework
- You should try a framework before dissing it
- The plethora of web frameworks is a good thing!
- Doing proper research can save time and money
- Testing is the best path to future maintenance
Questions?

matt@raibledesigns.com
http://raibledesigns.com

Download presentation from:
http://raibledesigns.com/rd/page/publications